EXHIBIT E

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent No: 7,714,747	**R.C.N.:** Not yet assigned
Issued: May 11, 2010	**Examiner:** Not yet assigned
Applicants: Fallon, James	**Art Unit:** Not yet assigned
Title: Data Compression Systems and Methods	

<u>REQUEST FOR *INTER PARTES* REEXAMINATION</u>

Mail Stop *Inter Partes* Reexam
Central Reexamination Unit
Office of Patent Legal Administration
United States Patent & Trademark Office
P.O. BOX 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

> After loading and unzipping the containers, the decompressor parses the structure container, invokes the corresponding semantic decompressor for the data items and generates the output.

(XMill at p. 9.)

selecting one or more lossless decoders for a data block associated with the data packet, wherein the selecting is based on the descriptor; *(part of claim 1)*

XMill discloses that the decompressor parses the structure container to select the semantic decompressor corresponding to the data items associated with the input data:

> After loading and unzipping the containers, the decompressor parses the structure container, invokes the corresponding semantic decompressor for the data items and generates the output.

(XMill at p. 9.)

XMill discloses a lossless configuration embodiment where white space is preserved:

> So far we have ignored white spaces between tags, e.g. between <Book> and <Title>, and the decompressor produces a standard indentation: this is sufficient for most applications. Optionally, XMill can preserve the white spaces faithfully: in that case it stores them in container 1.

(XMill at p. 10.)

decompressing the data block with a selected lossless decoder utilizing content dependent data decompression, if the descriptor indicates the data block is encoded utilizing content dependent data compression; and *(part of claim 1)*

XMill discloses that the compressor performs content-dependent encoding on data:

> **Apply different compressors to different containers** Some data items are text, others are numbers, while others may be DNA sequences. XMill applies different specialized compressors (semantic *compressors*) to different containers.

(XMill at p. 2.)

XMill discloses a structure container with information indicating the encoders used by the XMill compressor to compress the data; the XDemill decompressor parses the structure container to identify the corresponding decoders to use for decompression. If a content-

dependent semantic compressor was used to compress the data, a corresponding content-dependent decompressor will be used to decompress:

> After loading and unzipping the containers, the decompressor parses the structure container, invokes the corresponding semantic decompressor for the data items and generates the output.

(XMill at p. 9.)

decompressing the data block with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression. *(part of claim 1)*

XMill discloses that the compressor performs content-independent encoding on data by copying it to a container and then using a default gzip compressor:

> Users, of course, do not have to specify any semantic compressor: the default text semantic "compressor" simply copies its input to the container, without any semantic compression.

(XMill at p. 9.)

> The text compressor t does not compress, but rather copies the string to the container unchanged (it will be compressed later by gzip).

(XMill at p. 13.)

XMill discloses a structure container with information indicating the encoders used by the XMill compressor to compress the data. The XDemill decompressor parses the structure container to identify the appropriate decoders to use for decompression. If the structure container indicates that a data item was only copied by the text compressor without compression, then only content-independent unzipping decompression is used for that data item:

> After loading and unzipping the containers, the decompressor parses the structure container, invokes the corresponding semantic decompressor for the data items and generates the output.

(XMill at p. 9.)

Thus, as set forth in detail above, XMill teaches each and every feature of independent claim 1. Accordingly, claim 1 is unpatentable under 35 U.S.C. § 102(b) as anticipated by XMill.

> performance general-purpose coder, but the huge performance gains of schema-specific encoding are lost.

(XML-Xpress at p. 2.)

XML-Xpress discloses the use of lossless decoders:

> Lossless compression as high as 34:1 at throughputs of up to 9 Mbytes/sec were achieved on a test database.

(XML-Xpress at p. 1.)

> All coders were evaluated in lossless mode - the decoded field had to be a byte-for-byte match with the source files.

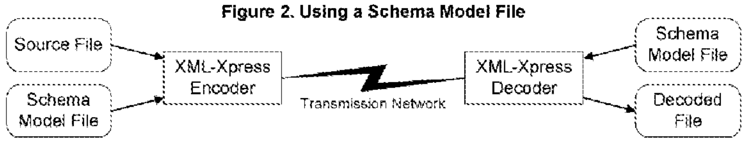
(XML-Xpress at p. 5.)

decompressing the data block with a selected lossless decoder utilizing content dependent data decompression, if the descriptor indicates the data block is encoded utilizing content dependent data compression; and

XML-Xpress discloses performing content-dependent data decompression based on the encoders identified by the schemas, if the schema indicates that the file was compressed using content-dependent data compression:

> Each SMF supports a specific schema. To compress an XML file, the XML-Xpress real-time encoder needs to have access to the SMF that supports the schema to which the file conforms

Figure 2. Using a Schema Model File



> The same SMF is provided to both the encoder and the decoder. The real-time encoder can support multiple schemas as long as it has the SMF for each and knows which one to use for each file. This information is available in most XML applications, so that they can take full advantage of the power of schema-specific coding.

(XML-Xpress at p. 3.)

> The schema allows the XML tags to be encoded with high efficiency. For example, if an element contains two sub-elements (A,B), the decoder can reconstruct the tags without needing any information from the encoded file: it knows exactly where the start and end tags of both elements A and B will be located.

> [K]nowledge of the schema allows the element data to be encoded efficiently. Because schemas supply the data type information, compression routines optimized for specific data types can be used.

(XML-Xpress at p. 2.)

XML-Xpress discloses the use of lossless decompression:

> Lossless compression as high as 34:1 at throughputs of up to 9 Mbytes/sec were achieved on a test database.

(XML-Xpress at p. 1.)

> All coders were evaluated in lossless mode - the decoded field had to be a byte-for-byte match with the source files.

(XML-Xpress at p. 5.)

decompressing the data block with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression. *(part of claim 1)*

XML-Xpress discloses performing content-independent data decompression using a generic algorithm, if the data block was encoded using the general-purpose, content-independent coder:

> If a file does not conform to the expected schema, the data is safely encoded using a high-performance ***general-purpose coder***, but the huge performance gains of schema-specific encoding are lost.

(XML-Xpress at p. 2) (emphasis added).

> For environments that support a large number of schemas, XML-Xpress can operate like ICT's flagship product XpressFiles - providing specialized support for the most common schemas and using a generic algorithm for the rest. In this way, the performance benefits of schema-specific coding are obtained within systems serving a diverse range of schemas.

(XML-Xpress at p. 4.)

XML-Xpress discloses the use of lossless decompression:

> Lossless compression as high as 34:1 at throughputs of up to 9 Mbytes/sec were achieved on a test database.

(XML-Xpress at p. 1.)

> All coders were evaluated in lossless mode - the decoded field had to be a byte-for-byte match with the source files.